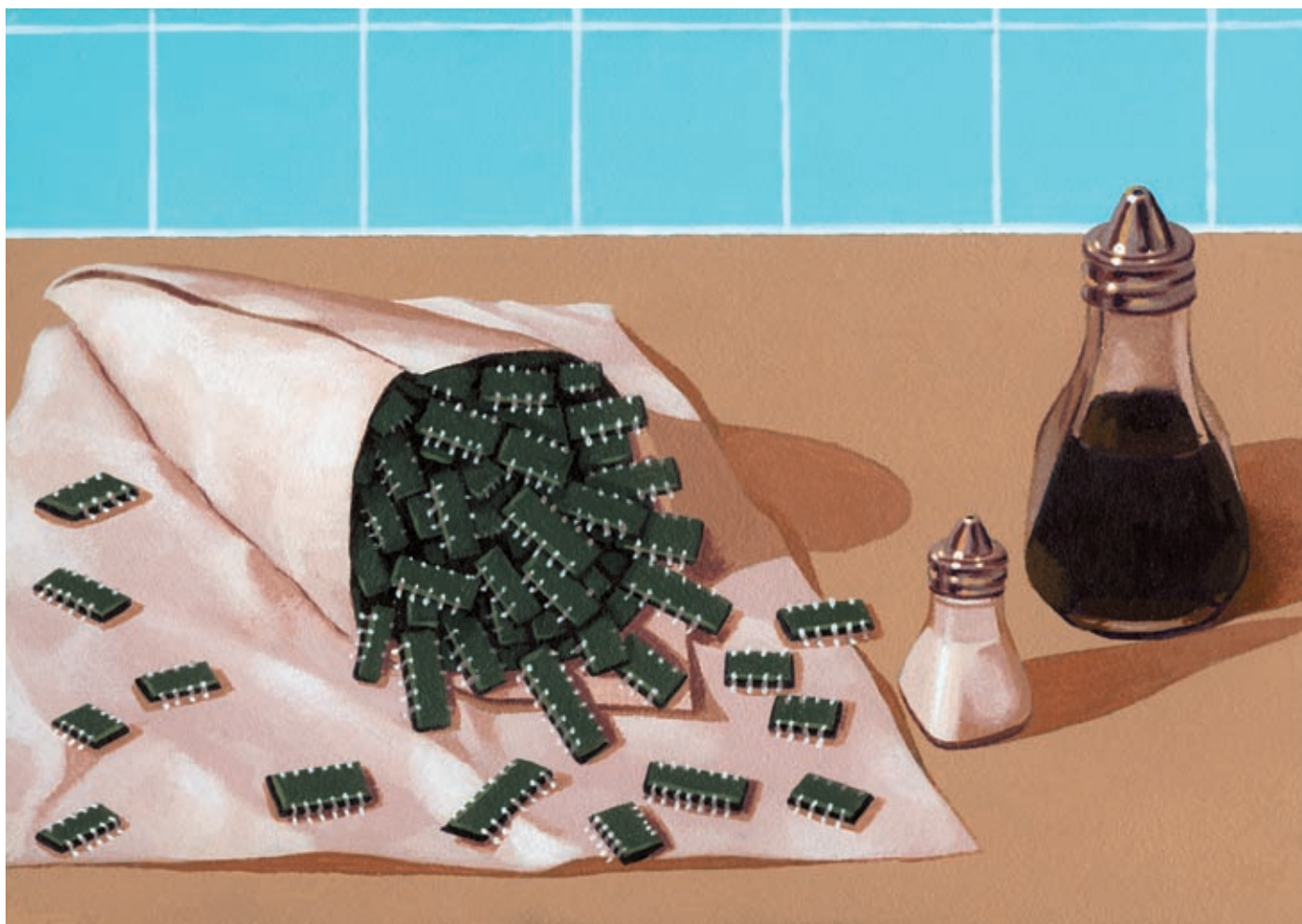




CHEAP AS CHIPS

You might think that a £50 AMD processor is cheap, but useful chips can be manufactured for as little as 20p. Mike Bedford investigates the microcontroller technology that gives household appliances their power

A top-of-the-range, dual-core Pentium processor will set you back nearly £800, so you'd be excused for thinking that an entry-level AMD Sempron chip, at less than £50, is a bargain. In the world of PCs a £50 processor is indeed cheap, but in the realm of embedded processors – chips that are used to power everything from washing machines to DVD players – processing power comes much cheaper. Here processors start at about 20p each in volume, which is less than a bag of crisps.

At first glance the technology behind embedded processors will look familiar to most *Computer Shopper* readers, but there are crucial differences in the way they are built, the language used to run programs on them and the way they work. We'll look at all these issues in this feature, plus the up-and-coming technologies that could drive prices even lower.

WHAT DO YOU GET FOR 20p?

A processor that costs well under a pound might be something of an eye-opener, but that's only half the story. Processors such as the Pentium, which are designed for use in PCs, can't do anything by

themselves. To do anything useful, all sorts of additional components, including a motherboard chipset and memory, have to be added to the computer. Not so the bottom-end processors we're looking at here. These ultra low-cost components are called microcontrollers rather than microprocessors, but it's not just a difference in terminology. Microcontrollers include the processor, memory, and input/output circuitry on a single chip so you get even more for your money.

Before you get too excited about the prospect of tomorrow's PCs being given away in packets of cornflakes, we ought to put things in perspective. A few facts and figures will put these low-cost microcontrollers into context and convince you that you really wouldn't want one in your PC.

At the bottom end, microcontrollers might have a maximum clock speed of 4MHz and the onboard memory could total no more than 400 bytes. The clock speed is equivalent to that of the very first PC, at about 1,000 times slower than today's latest systems, and the memory is considerably less than you'd expect in an entry-level PC. What's more, because microcontrollers aren't used with hard disks, most of that memory is non-



volatile memory used to store the program. Genuine RAM might be no more than 20 bytes, or 20 billionths of a gigabyte. If that's not enough, bargain basement microcontrollers have an 8-bit architecture, which means they can work only with numbers in the range 0 to 255. Even the first PC used a 16-bit processor and today's most powerful processors employ a 64-bit architecture.

We'll see how manufacturers manage to design processors – even such basic ones – that sell for a few pence. But designing from scratch isn't the only way to produce budget processors. Intel's 8051 and 8052 microcontrollers are based on much the same technology as the 8080 and 8085 processors that were used in some of the home computers of the late 1970s and early 1980s.

Using the technology of two or three decades ago can take you a few steps up the ladder from a really basic microcontroller. Intel's products might cost a pound or so each, rather than a few pence, but you get rather more computing power for your money. At this sort of price – and we're now talking of mid-range microcontrollers – you might get 8K of non-volatile memory, 256 bytes of RAM and a 16MHz clock.

HOW ARE THEY USED?

If you were to use a microcontroller in a PC, even a mid-range microcontroller such as the Intel 8051, you'd end up with something not much more powerful than the Sinclair ZX80 home computer that took the world by storm over 25 years ago. Needless to say, PCs aren't where you'd find a low-end microcontroller. Instead, they're used for embedded applications, to use the industry jargon.

An embedded application is one in which the microcontroller is hidden away from the user. When you use your PC to write a letter or surf the web, you'll be aware that a Pentium 4 is working in the background. When you turn the ignition key in your car it might not be as obvious, but dozens of microcontrollers will start up. Some will be managing the instruments on the dashboard, others might be handling the lights and others will be involved with more advanced systems such as ABS and traction control.

It will come as no surprise that features such as adaptive cruise control and intelligent automatic transmission rely on digital technology. Indeed, their introduction relied on the availability of low-cost computing power. But cars have always had lights and windscreen wipers, and these low-tech systems can certainly be made to work with conventional electrical and electronic components. Yet if microcontrollers are cheap enough, they can be used in place of a handful of electronic components, resulting in fewer components, greater reliability, and more flexibility.

When we think about this sort of application it's easy to see why a processor clocked at 4MHz and with very little memory will suffice. The main reason that PC software is so memory-hungry is that it has to operate under Windows to provide an attractive-looking display. Without this constraint, an amazing amount of functionality can be built into a very small program and it's not hard to appreciate how the simple rules needed to control a car's lights can be crammed into just a few hundred bytes.

Even that rather pedestrian 4MHz clock speed (and 1µs instruction time) isn't an issue. If we assume that it takes 100 instructions to recognise

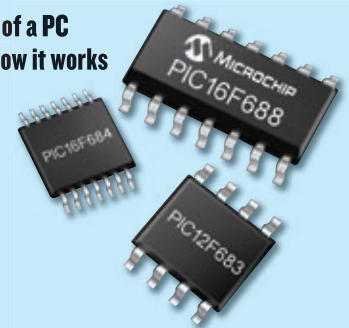
The world's cheapest microcontroller

The PIC 10F200 doesn't have much of the sophistication of a PC processor, but that simplicity means you can easily see how it works

If you like to keep your finger on the pulse of technology and you like to know how things work, you'll probably have looked at the diagrams of Intel's latest processors. You'll probably also have wondered – like we do when looking under the bonnet of a modern car – how all those pieces work together to provide top-end performance. Microcontrollers are different. Gone are the multi-stage pipelines, the multiple execution units, the multi-level caches, out-of-order execution and the like. Here is a piece of silicon that is simply designed to do a simple job, and it's a good place to learn something about processor design.

The diagram below shows the PIC 10F200, the world's simplest and cheapest microcontroller. The program counter is a single word of memory that contains the address of the next instruction to be executed. When the microcontroller is first switched on, it contains the address of the first instruction in the program memory.

Normally it's incremented after each instruction is executed so it points to the next instruction, but there are some exceptions. If the instruction is a jump, for example, the program counter is loaded with a new value via the data bus. Another exception regards calls to subroutines where the stack comes into play. Comprising two slots of last-in-first-out memory, the stack is used to remember the contents of the program counter before a subroutine call is executed so this address can be put back into the program counter when the subroutine terminates.

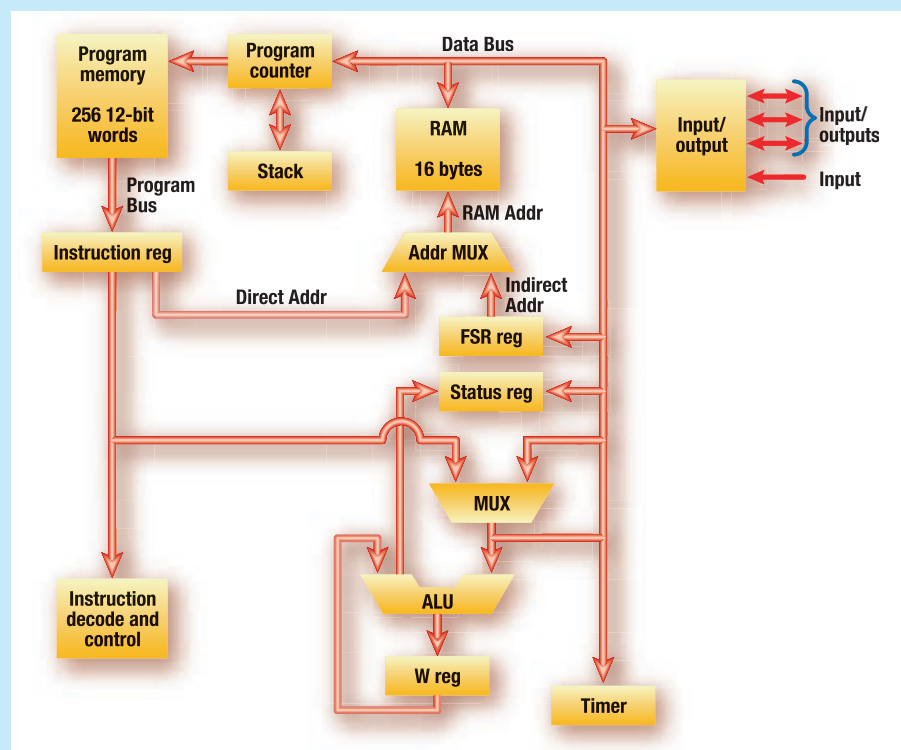


▲ Microcontrollers such as the ones above are used in everything from cars to washing machines

With each tick of the internal clock, the program counter is used to load a new instruction from the program memory into the instruction register. The instruction is interpreted in the instruction decode and control circuitry and any data required by the instruction is obtained from the block of RAM. Now the decoded instruction and its data are combined in a multiplexer (MUX) before being sent to the arithmetic and logic unit (ALU) for execution.

Depending on the instruction, the ALU will update the status register (which can affect the operation of subsequent instructions), the program counter and, perhaps, the W register, a special working register. It might also write a value to a location in RAM, communicate with the timer or communicate with the input/output circuitry. This drives the four I/O pins, one of which is an input and three of which are programmable. These connect to switches, sounders, indicators or other bits of external hardware.

The PIC 10F200 in detail





Designing and manufacturing a chip that sells for just a few pence requires a 'back to basics' approach in the factory

that the car door is open and turn on the courtesy light, at 4MHz there would be a delay of just a tenth of a millisecond. This is far faster than is really needed, so for this sort of application designers might choose to clock the processor even more slowly to save power.

THE MAKING OF A 20p PROCESSOR

It's not hard to appreciate that entry-level microcontrollers will be a fair bit cheaper than an entry-level PC processor from Intel or AMD. Even so, a price of 20p is quite something. With a bottom-end microcontroller containing perhaps 30,000 transistors, that's a grand total of 0.0007p per transistor, which is cheap by anyone's standards. So exactly how do semiconductor manufacturers manage to produce such a powerful bit of hardware for so little?

Douglas Chaffee is the director of PIC development engineering at Microchip Technology,

producers of the world's smallest microcontroller. "The microcontroller developer is faced with the challenge of embodying this wide range of functionality in a device that must conserve silicon area in order to keep cost at a minimum. To obtain this goal, the device designer must be willing to take the time required to optimise the circuit design. Often this requires the use of multiple chip design techniques, including full manual design and layout of critical circuits," he said.

To the uninitiated this might sound like a formula for designing any chip, but

apparently this is not the case. To make this point, Chaffee contrasts this approach with that used to design a processor for a PC. "In today's world of logic synthesis and automatic layout design tools," he said, "these techniques may seem ancient, but in the world of low-cost microcontrollers they are often the only path to the low cost that the market demands. The development of PC processors offers much higher functionality and computing power and is driven by time to market."

Chaffee suggests that "this, coupled with the constantly shrinking silicon geometries, allows the developer to utilise the more advanced design techniques of synthesis and automatic place and route. These techniques offer a shorter, more predictable design cycle for these highly complex devices. The microcontroller's low cost requirements, on the other hand, demand that the developer sacrifice speed of development for silicon efficiency."

HOW ARE THEY PROGRAMMED?

So much for designing a low-cost microcontroller, but what about the job of the software engineer tasked with using one in a piece of electronic equipment? Cars and washing machines don't run Windows, so it will be quite different from programming for a PC, but what type of operating system is used and how are they programmed?

Steve Diaper, field applications engineer with Microchip Technology, explains. "With the limited resources of an 8-bit microcontroller, an operating system is a luxury few applications can afford. Only when applications demand higher-end 8-bit or 16-bit microcontrollers with larger memory does the use of any sort of operating system become viable.

"In contrast to the PC market, where the growth of disk space, memory and processing capability seems at least partly driven by the need to support more complex operating systems, rather than providing more computing power to the user, the focus in embedded system development is to do more with less."

To those familiar with programming PCs, having to make do without an operating system and all the facilities that it provides will be something of an eye-opener. So what about the programming language? C, C++, Java, perhaps? Apparently not. According to Diaper, "the vast majority of applications are still programmed in assembly language. As with operating systems, a high-level language becomes more important in larger 8-bit and 16-bit applications. Developers should bear in mind, though, that using a high-level language will have somewhere in the region of a 30 per cent code overhead versus an assembly language development."

Diaper further explains that using assembler language means you are working more closely with the architecture and I/O of the device you are programming. "Even if the decision is made to use a high-level language, the developer still needs a good appreciation of the hardware in the system.

How to... write your own microcontroller program

Microcontrollers are normally programmed in assembler instead of a high-level language and the programs are not designed to run under an operating system. So if you're familiar with writing programs for a PC, working with a microcontroller couldn't be more different. However, if you fancy trying your hand at writing some microcontroller code it's really not too difficult – as long as you understand the basics of programming – and it needn't cost much money.

To write your software you need a so-called development environment. These are available for most microcontrollers and run on a PC. They provide a means of entering and editing your software and an assembler that checks for errors.

As long as your code passes the test, the assembler compiles it to produce executable code for the microcontroller. You then have to transfer it into a microcontroller's non-volatile memory using a piece of hardware called a programmer, which connects to your PC via a serial or USB port.

But getting the code into a microcontroller is only half the story. The microcontroller has to be

connected to a power supply and, since your program will be assuming some external hardware – perhaps a few buttons and LEDs – you'll need to connect these, too. If you're handy with a soldering iron you could do this yourself, but it's far easier to get an evaluation kit. This includes a microcontroller, a power supply, and various switches and indicators. It might also include a programmer on the same board so you don't have to unplug the microcontroller to program it and then plug it back in to try out your code.

The PIC microcontrollers from Microchip Technology would be a good choice to get a feel for the technology. The development environment, MPLAB IDE v7.22, is a free, though large, download from www.microchip.com. It even has a simulator that will let you see how your program will behave, even if you don't have the PIC hardware.

If you want to see your code running on a real microcontroller you should buy the PICkit 2 Flash Starter kit. This is supplied with the development software and seven tutorials, each accompanied by sample code, which you can either use as they

stand or as a starting point for your own projects. You can buy the kit (part number DV164101) from <http://buy.microchip.com/productsearch.aspx?keywords=dv164101> for £23 including VAT.



▲ The PICkit 2 Flash Starter kit from Microchip Technology is a good way to try out microcontroller technology for yourself on your PC. It costs just £23 including VAT



As an example, when developing in C on a PC, the 'printf' instruction is very standardised, and prints the text string that follows on to the monitor of the PC."

However, with an embedded system things are not nearly so simple, as he points out. "So, how should a printf instruction operate within a microcontroller application?" he queried. "It could print the string on an LCD screen attached to the PIC microcontroller – and that would require the appropriate hardware driver to do so – or in the absence of an LCD screen, it could output the string to a serial port on the PIC microcontroller. So, as you can see, even programming in C is very different in an embedded environment."

EVEN CHEAPER CHIPS

Making chips out of silicon is an expensive business and sometimes even 20p is too much to pay for intelligence. But plastics are cheap to work with, so researchers are developing electronic circuitry made out of polymers to drive down costs even further. How far has this research come and when is it likely we'll see the first plastic processor?

Duncan Barclay, group leader for logic, RFID and electronics at polymer electronics company Plastic Logic, explains that the reason why plastic is so much cheaper is "to do with the manufacturing techniques used. Plastic electronics can be made with processes very similar to those used in the printing industry – in fact, we make plastic electronics using inkjet printers. The capital equipment cost of such a plastic electronics printing plant will be many times less than a silicon fabrication plant, so the cost carried in each plastic chip to recoup the capital costs will be much less."

So is this already a practical technology or just an idea for research? "A few products exist; however, there is not yet a high-volume commercial product using plastic electronics." And are any of those products microcontrollers? "To the best of my knowledge, no-one has demonstrated a microcontroller made from plastic electronics to date," Barclay admits. He explains how, although it could be done, the transistors would be so large that it would be the size of a credit card and have the performance of the 4004, the world's first microprocessor.

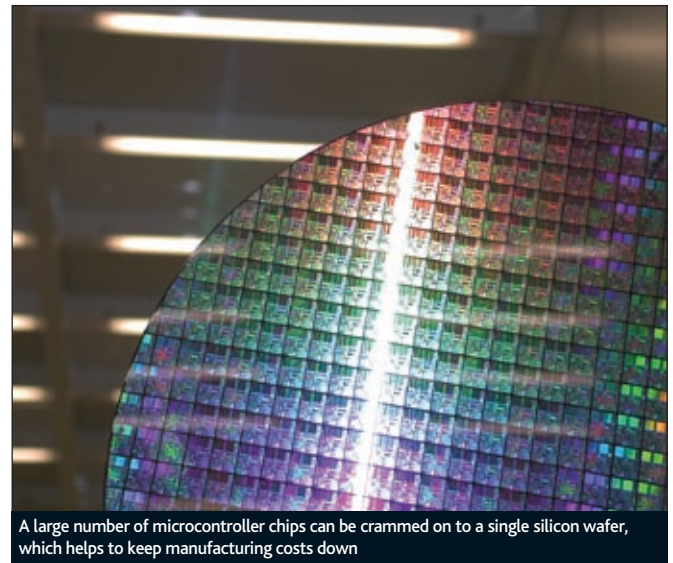
So when, if ever, are we likely to see plastic processors or microcontrollers? Here Barclay is more upbeat. "I would expect that research labs will produce a micro in the next two to three years." And if his price comparison – that plastic chips could be 20 to 30 times cheaper than silicon chips – is accurate, that will be a seriously cheap processor.

THE FUTURE

The statement attributed to IBM founder Thomas J Watson in 1943 that there may be a world market for five computers is now legendary. With over 820 million PCs around the globe, he was clearly wrong. But compared to the market for

microcontrollers, this is nothing. It's estimated that the average car contains between 25 and 35 microcontrollers and the average household even more. In coming years, though, even this might prove to be just the tip of the iceberg.

If plastic processors ever reach their potential, they'll be everywhere. They'll be so cheap that microcontrollers will even be used as intelligent barcodes, printed on the packaging of our groceries and communicating with our intelligent fridges. In the not-too-distant future we could each be throwing away more computing power every day than Thomas Watson envisaged there might ever be in the whole world. **CS**



A large number of microcontroller chips can be crammed on to a single silicon wafer, which helps to keep manufacturing costs down

HOW IT WORKS

Plastic Electronics

Metals are electrical conductors; plastics are electrical insulations. That might be the conventional thinking, but times are changing and scientists have now made plastics that can conduct electricity and are using them to make electronic components. Here we see how the laws of chemistry have been turned on their head to bring us the promise of even cheaper processors.

An electric current involves the flow of electrons. Some of the electrons in a piece of metal are free to move around, rather than being tied to individual atoms, which is why they are able to conduct. Most organic substances, including polymers – the technical name for plastics – have strong covalent bonds between the atoms. This means that the electrons are fixed, which is bad news for electrical conduction.

However, there are exceptions, one of which you may remember from chemistry lessons. The benzene molecule comprises six carbon atoms in a ring with a hydrogen atom bonded to each. This is often shown with alternate single and double chemical bonds between the carbon atoms in the ring, but this isn't an accurate representation. Instead, some of the electrons in those bonds are referred to as delocalised. They're not associated

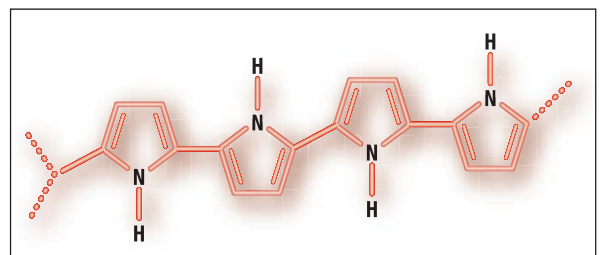
with particular atoms so they are free to move around the ring.

At first sight this might sound like a recipe for electrical conduction, but it's not quite that simple. Those delocalised electrons are free to move only within a single molecule; they can't jump from one molecule to another, so benzene is still an insulator.

Polymers are different. Whereas the benzene molecule contains 12 atoms, and other organic molecules may contain a few hundred, polymers have vast molecules containing small groups of atoms repeating over and over again. So, if scientists can devise a polymer that has delocalised electrons, like benzene, it would allow electrical conduction across an appreciable distance.

The diagram on the right shows the chemical structure of Polyphenyl-vinylene (PPV), which was one of the first conducting plastics to be synthesised. This is only part of a molecule; as indicated by the dotted lines, the chain extends in both directions almost indefinitely. It's referred to as a conjugated polymer, which

means that it has a chain of alternate single and double chemical bonds. As with the benzene molecule, though, this is an oversimplification. In practice, some of the electrons are delocalised, which gives rise to electrical conduction. The basic polymer is purely a conductor but by modifying the structure slightly – by adding additional groups of atoms off the main chain – it's possible to alter the electrical properties to get a semiconductor. And if you have ordinary insulating plastics, conducting plastics and both p-type and n-type semiconductor plastics, you have all the building blocks of a microcontroller.



The chemical structure of Polyphenyl-vinylene was one of the first conducting plastics. Its 'delocalised' electrons give rise to electrical conduction